Simulated Railroad Framework, http://simulrr.sourceforge.net
Synopsis: 000_Synopsis

This file valid for step 0033.10
Issue Date: 2017-04-01

The SMS Tracer

1 Synopsis

It is the intention to use the SMS tracer for
- getting familiar with SRR/SMUOS software
- debugging SRR/SMUOS software
- documenting SRR/SMUOS software

2 Purpose

The programmer can write diagnostic output to the tracer, which will be
- output to the browsers console
- output at the uiControl interface

The second option is intended to process trace output by external programs via the SAI/EAI.

Each place in the code, where diagnostic output is sent to the tracer - we call these places tracepoints - has assigned a trace level.

Only if the actual trace level of the system is equal to or greater than the trace level of the tracepoint, then the diagnostic output will actually be written.

Different parts of the system can have different trace levels, enabling narrowing down the error/effect you're looking for.

2.1 Trace Levels

0......no tracer output
1......Errors (this trace level is the default setting in the system)
2......Infos
3......Debug Infos

3 External View (Using the Tracer for Debugging and Getting Familiar)

3.1 The Structure of the System

The multi user session is built by a MU server and one or several scene instances.

Note: The current tracer can only be used on the scene instances but not on the MU server.

Each network sensor of the SRR/SMUOS Framework connects clients with a server, this can be depicted as in the following example:

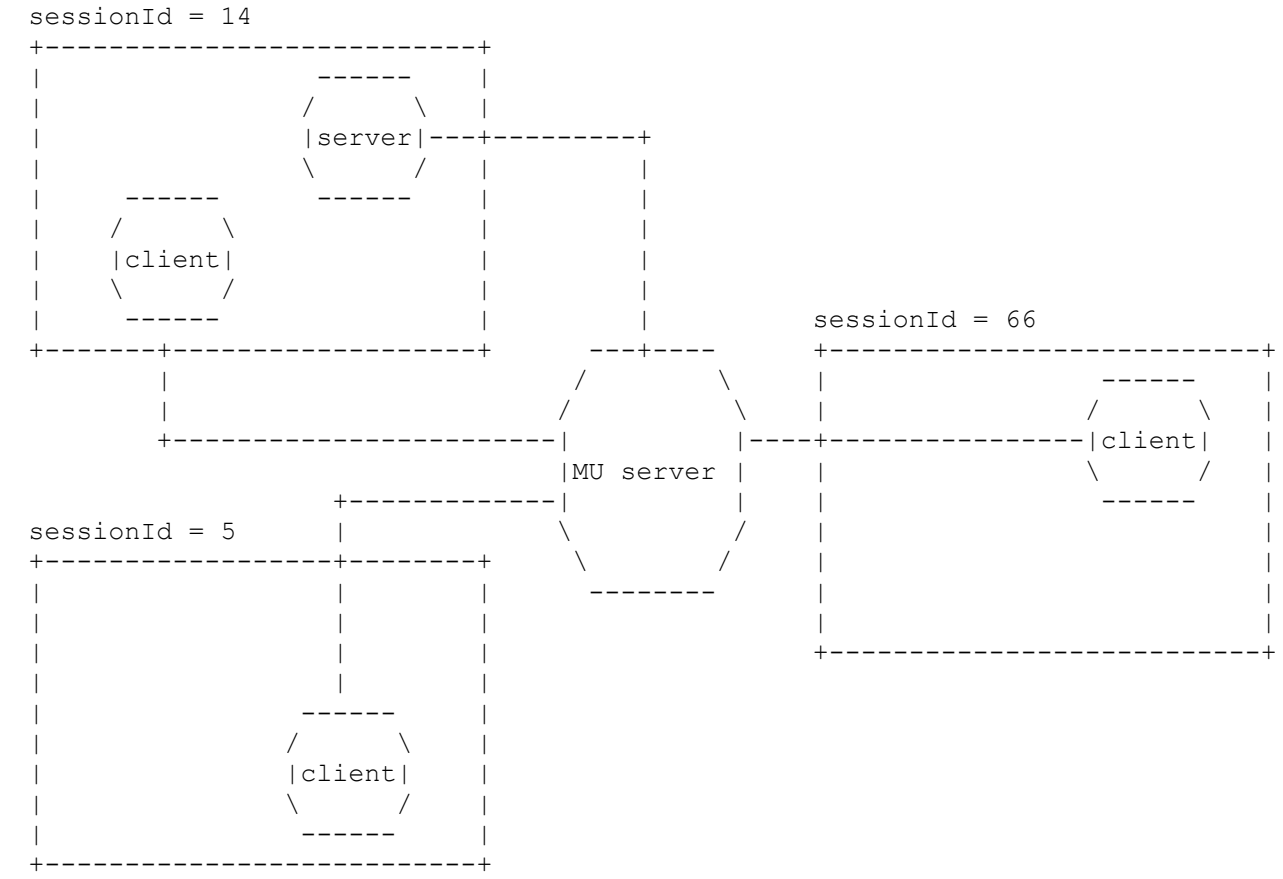

Figure 1: Clients, Servers and Scene Instances (scope of one network sensor)

Network Sensors are currently used in the Simple Scene Controller and in the MIDAS Objects, the Module Coordinator has currently no server software and hence no network sensor. On the other hand, the Simple Scene Controller contains one SSC dispatcher for each registered module and one for each universal object class (UOC), so that tracer instances are necessary for module/UOC "clients" and "servers".

See following table for an overview:

Part of the Framew.	Client instance	Server instance
Simple Scene Contr.	SscBase	CommControl
	SscBeamerManager	
	SscKeyManager	KeyControl
	SscTrainManager	TrainControl (not yet impl.)
SSC Dispatcher	<dispatcherName>.Client	<dispatcherName>.Server
Module Coordinator	<m>*-McBase.Coord	n/a (not yet used)
	<m>*-TrainManager.Coord	
MIDAS Object Dispatcher Stub	<extObjId>.Control	<extObjId>.ObCo
	<extObjId>.Server	

Table 1: Names of Tracer Instances

* <m> stands for a <moduleName>

3.2 The Structure of a Scene Instance

A scene instance consists of Models, Modules and a Frame. Each of these parts use distinct parts of the SRR/SMUOS Framework:

- Models use MIDAS Objects and are attached to modules.
- Each Module uses an instance of the Module Coordinator.
- The Frame uses the Simple Scene Controller

Hence the structure of an example scene instance can be depicted as follows:

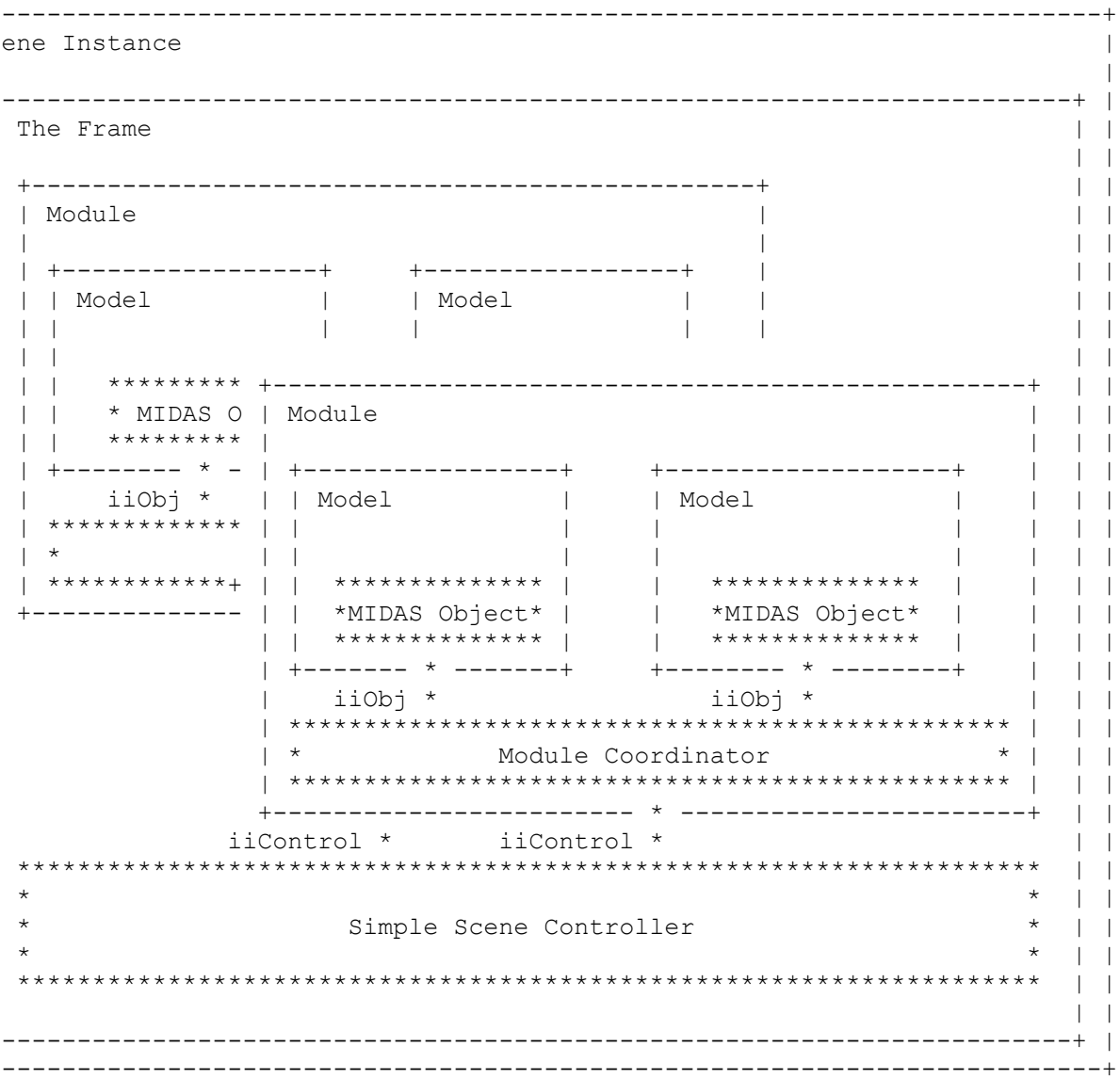

Figure 2: Structure of a Scene Instance

3.2.1 The SRR/SMUOS Framework has got internal interfaces

Internal Interface	within
iiControl	Simple Scene Controller
miControl	minimum interface of an SSC Extension towards SSC Base
iiDisp	Simple Scene Controller
miMod	minimum interface of an MC Extension towards MC Base

3.2.2 The SRR/SMUOS Framework has got external interfaces (towards programmers)

External Interface	provided by
eiControl	Simple Scene Controller to MC and MIB
eiMod	Module Coordinator to MIB
eiMib	MIDAS Base (MIB) to MIDAS Objects
eiConsole	Simple Scene Controller to MIDAS Objects

3.2.3 The SRR/SMUOS Framework has got user interfaces (towards authors)

External Interface	provided by
uiControl	Simple Scene Controller to Frame
uiMod	Module Coordinator to Module
uiObj	MIDAS Object to Model/Module/Frame

3.2.4 The SRR/SMUOS Framework defines minimum interfaces (for authors)

Minimum Interface	provided by
miModule	Module to Frame
miModel (bound model)	Model to Module
miModel (unbound model)	Model to SRR/SMUOS Framework

3.3 Setting the Trace Levels

Trace levels are set locally in one scene instance and must hence be set in each required scene instance separately.

The trace levels can be set at the uiControl interface with the following input fields of the Simple Scene Controller.

3.3.1 traceLevelRequest (SFInt32)

This input field sets the "classic" trace level. The "classic" tracer is not used by the SRR/SMUOS Framework and is hence not described here.

3.3.2 traceLevelSscBaseRequest (MFInt32)

This input field sets the trace level of the tracer instance "SscBase" (i.e. of the clients of the SSC Base module and of some SSC extensions that follow SscBase).

The input field reacts on the first two elements of the MFInt32 array, the trace level [0] will be applied AFTER initialization ("operational" trace level) and the trace level [1] will be used during initialization ("initialization" trace level).

The switchover between the two trace levels happens, when the SSC Base issues the "common parameters" (commParam), i.e. BEFORE the module coordinators are initialized.

3.3.3 traceLevelCommControlRequest (MFInt32)

This input field sets the trace level of the tracer instance "CommControl" (i.e. of the server of the SSC Base and of some SSC extensions that follow the SscBase).

The input field reacts on the first two elements of the MFInt32 array, the trace level [0] will be applied AFTER initialization ("operational" trace level) and the trace level [1] will be used during initialization ("initialization" trace level).

The switchover from "initialization" trace level to "operational" trace level happens, when the SSC Base issues the "common parameters" (commParam), i.e. BEFORE the module coordinators are initialized.

When you want to trace the CommControl instance, be sure to have the "central controller role" in your scene instance!

3.3.4 traceLevelModulesRequest (SFString)

This SFString value requests the trace levels for all(!) modules in a simple syntax.

Just request
   <moduleName>=<tloper>[,<tlinit>][;<moduleName>=<tloper>[,<tlinit>]]...

for as many modules as you like, where
   <moduleName> is the name of the module in question ('*' as a place holder is allowed)
   <tloper> is the "operational" trace level
   <tlinit> is the "initialization" trace level.

If you omit <tlinit>, it will be set to <tloper>.

The switch over from "initialization" trace level to "operational" trace level happens, when the module coordinator issues the "module parameters" (modParam), i.e. BEFORE the MIDAS Objects are initialized.

The SRR/SMUOS Framework will automatically add a leading term "*=1,1;" to set the trace level of all modules that you do not specify.

3.3.5 traceLevelObjectsRequest (SFString)

This SFString value requests the trace levels for all(!) objects in a simple syntax.

The syntax and logic is similar to (3.3.4), but <dispatcherName>-<objId> is used instead of <moduleName> and only one trace level is used (<tlover>, the "overall" trace level").

When you want to trace the <dispatcherName>-<objId>.ObCo instance of a MIDAS Object, be sure to have the "MOC role" for the parent/current module in your scene instance!

4 Internal View (Using the Tracer as Programmer)

tbd.

5 Additional Info

none